SynthsGW

COP 4331 Presentation - Group 6

Jonathan Brownlee
William Young
Leo Crespo
Morgan Cole
Dylan Ortiz

What SynthsGW is

Our software is for the amateur musician interested in creating custom synths and beats in an interface which allows them to upload their own songs, edit their songs, and incorporate the synths and beats to go along with their song. This is for musicians who have some experience with gathering their music, but our software allows users to categorize moments of music, too.

Our Mission

- SynthsGW strips down all of the mechanics of popular music making software and gives the user to add beats, create synths, and use existing mp3 sounds
- Users can use the built in Synth to create a new basic track of their own and work on it
- SynthsGW isn't trying to recreate anything, it’s giving the user the ability to create custom songs with just a few tools

User Stories - As a user I want to be able to…

- Create custom beats to make my own sounds for music
- Edit a beat so that i can make changes if I want
- Play beats back so that i can hear what I’ve made
- Edit a synth once it is done so that I can make changes if I want
- Get help with the software by going to the GitHub page
- See the different synths, beats, mp3s that are playing so I can control my sounds
- Record my synth so I can produce music

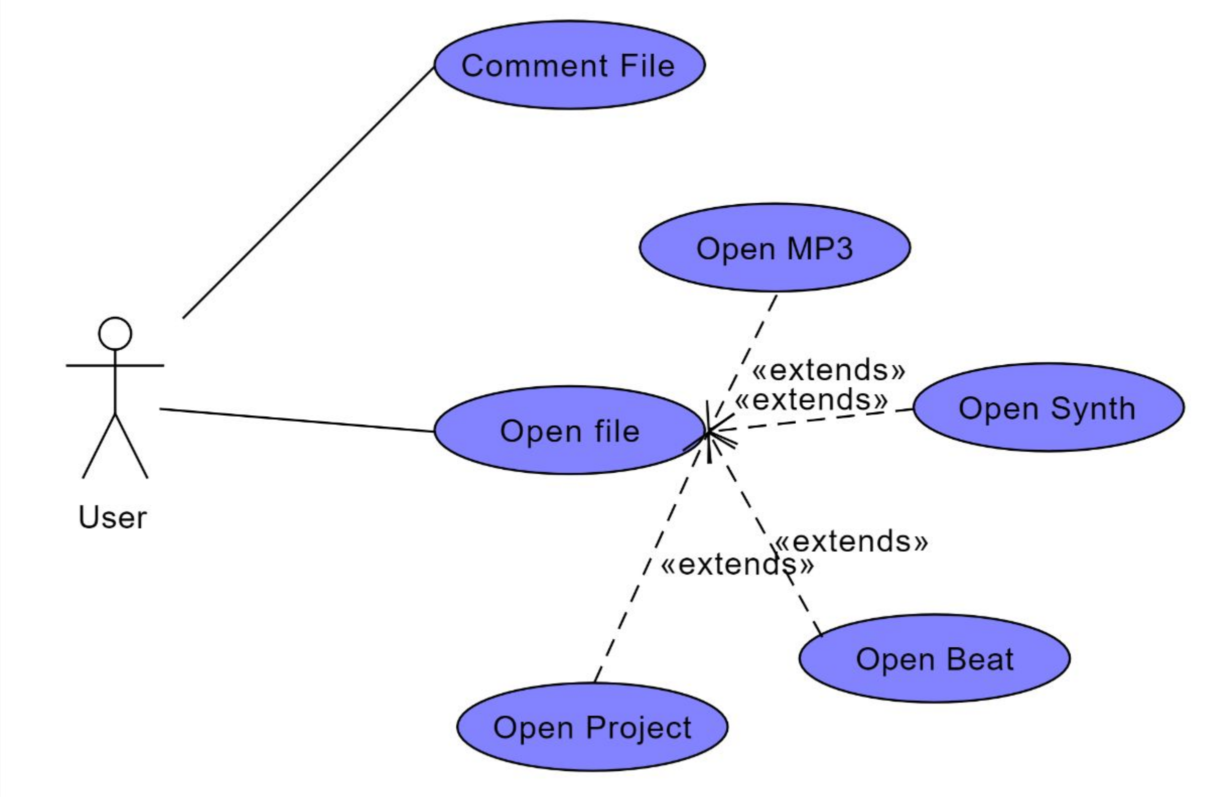
Requirements - The software shall be able to...

- Store beats in a directory of the user’s choice
- Name a beat file
- Edit a beat
- Play beats back
- Store synths in a directory of the user’s choice
- Name a synth file
- Close an open file
- Show the different files playing

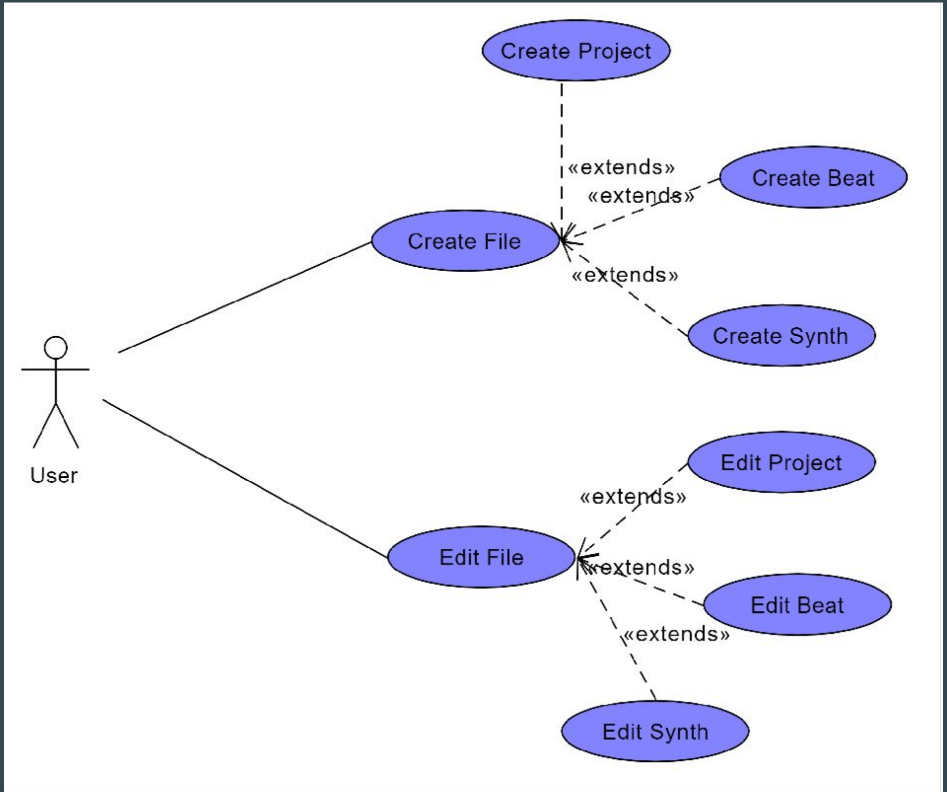
System Design

- Following a MVC architecture
- Java 8
- JavaFX w/ Scenebuilder

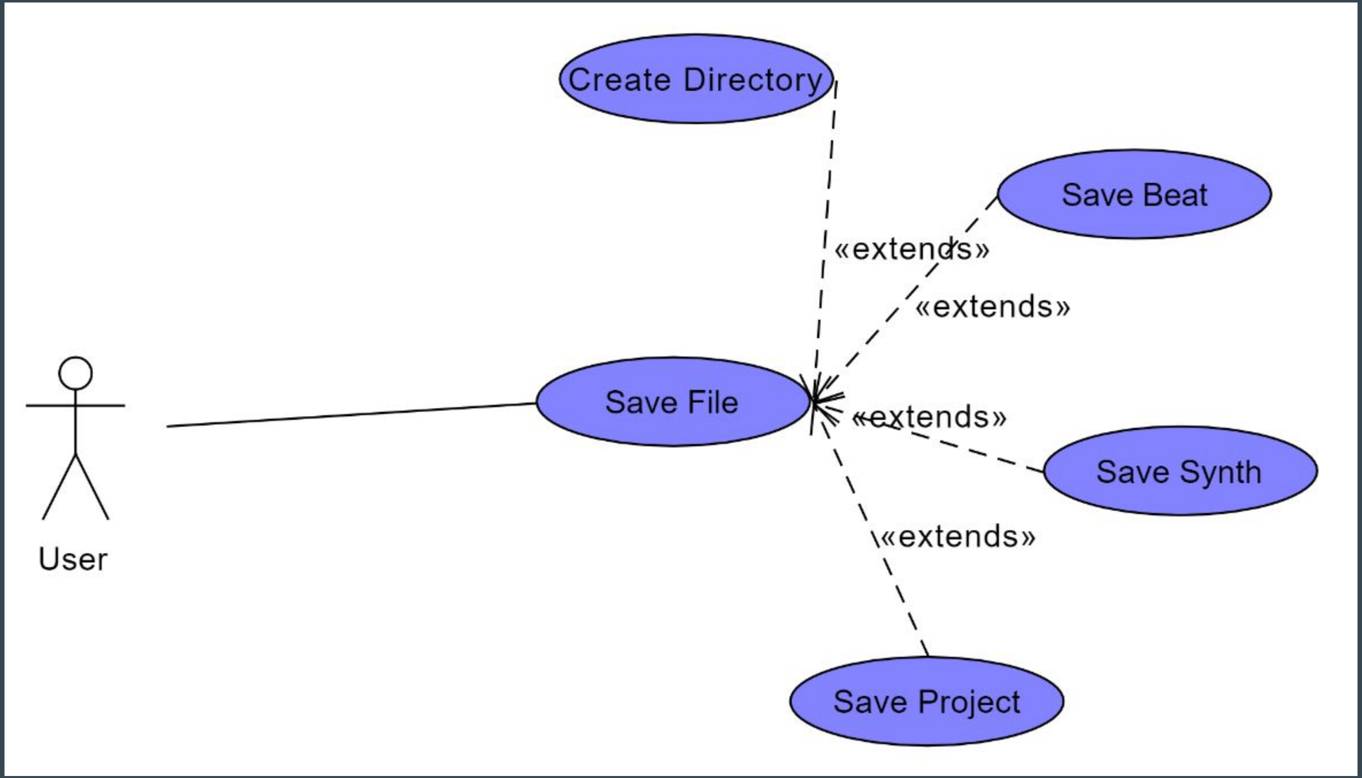
Use Case Diagram 1



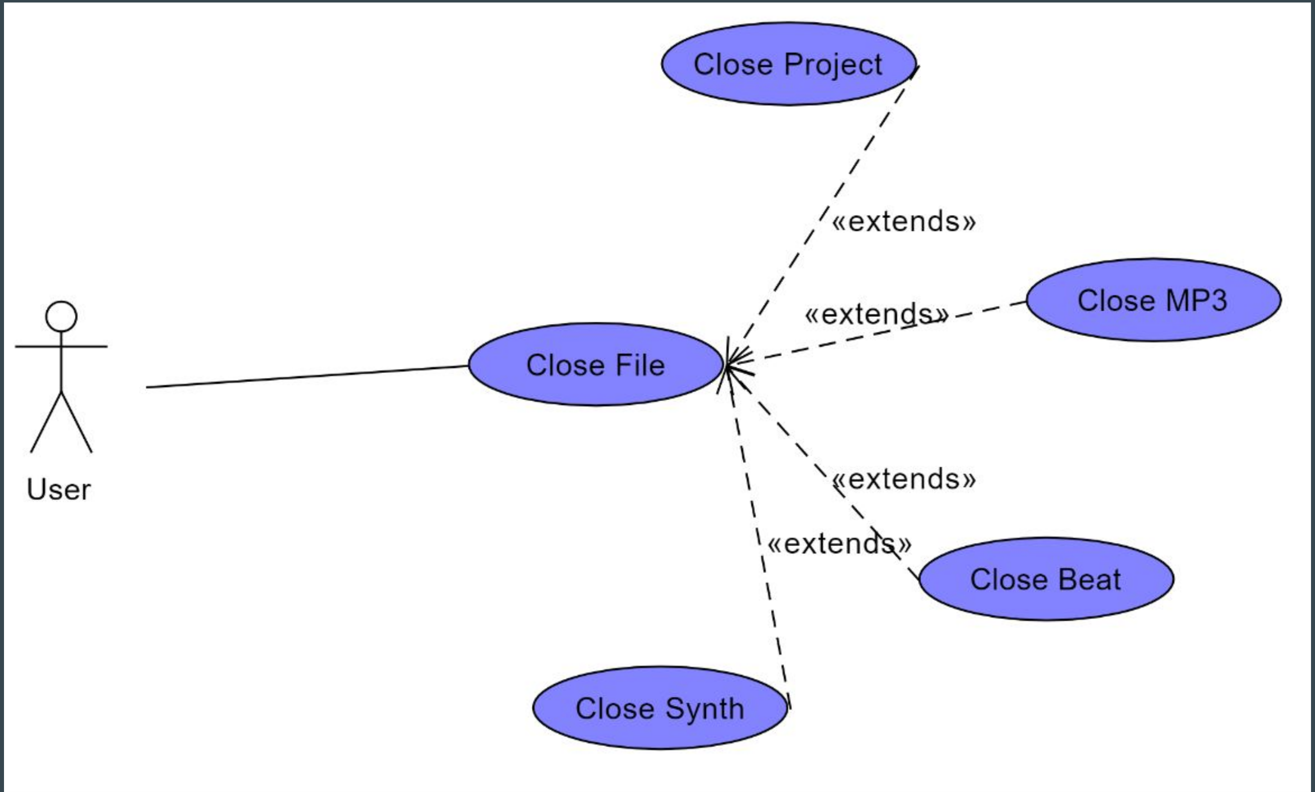
Use Case Diagram 2



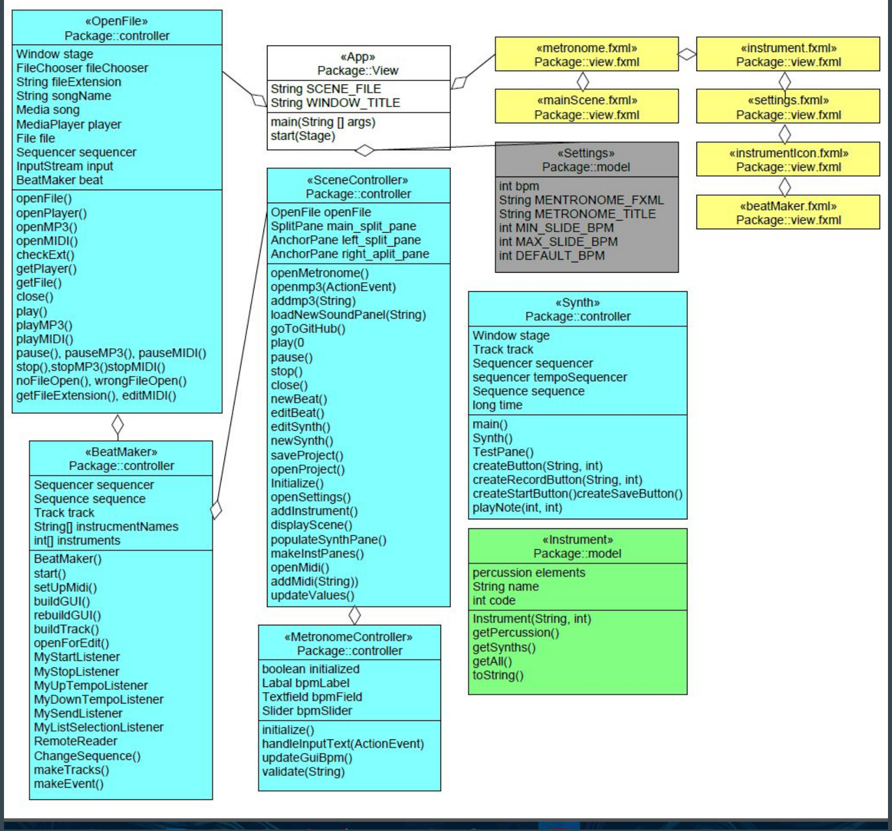
Use Case Diagram 3



Use Case Diagram 4



Class Diagram



«OpenFile»
Package::controller

Window stage
FileChooser fileChooser
String fileExtension
String songName
Media song
MediaPlayer player
File file
Sequencer sequencer
InputStream input
BeatMaker beat

openFile()
openPlayer()
openMP3()
openMIDI()
checkExt()
getPlayer()
getFile()
close()
play()
playMP3()
playMIDI()
pause(), pauseMP3(), pauseMIDI()
stop(),stopMP3()stopMIDI()
noFileOpen(), wrongFileOpen()
getFileExtension(), editMIDI()

«BeatMaker»
Package::controller

Sequencer sequencer
Sequence sequence
Track track
String[] instrucmentNames
int[] instruments

BeatMaker()
start()
setUpMidi()
buildGUI()
rebuildGUI()
buildTrack()
openForEdit()
MyStartListener
MyStopListener
MyUpTempoListener
MyDownTempoListener
MySendListener
MyListSelectionListener
RemoteReader
ChangeSequence()
makeTracks()
makeEvent()

«App»
Package::View

String SCENE_FILE
String WINDOW_TITLE

main(String [] args)
start(Stage)

«SceneController»
Package::controller

OpenFile openFile
SplitPane main_split_pane
AnchorPane left_split_pane
AnchorPane right_aplit_pane

openMetronome()
openmp3(ActionEvent)
addmp3(String)
loadNewSoundPanel(String)
goToGitHub()
play(0
pause()
stop()
close()
newBeat()
editBeat()
editSynth()
newSynth()
saveProject()
openProject()
Initialize()
openSettings()
addInstrument()
displayScene()
populateSynthPane()
makeInstPanes()
openMidi()
addMidi(String))
updateValues()

«MetronomeController»
Package::controller

boolean initialized
Labal bpmLabel
Textfield bpmField
Slider bpmSlider

initialize()
handleInputText(ActionEvent)
updateGuiBpm()
validate(String)

«metronome.fxml»
Package::view.fxml

«mainScene.fxml»
Package::view.fxml

«Settings»
Package::model

int bpm
String MENTRONOME_FXML
String METRONOME_TITLE
int MIN_SLIDE_BPM
int MAX_SLIDE_BPM
int DEFAULT_BPM

«instrument.fxml»
Package::view.fxml

«settings.fxml»
Package::view.fxml

«instrumentIcon.fxml»
Package::view.fxml

«beatMaker.fxml»
Package::view.fxml

«Synth»
Package::controller

Window stage
Track track
Sequencer sequencer
sequencer tempoSequencer
Sequence sequence
long time

main()
Synth()
TestPane()
createButton(String, int)
createRecordButton(String, int)
createStartButton()createSaveButton()
playNote(int, int)

«Instrument»
Package::model

percussion elements
String name
int code

Instrument(String, int)
getPercussion()
getSynths()
getAll()
toString()

Technology

- The functionality and some of the GUI elements were made using Java 8
- The main GUI Window along with the pop up windows were made using JavaFX and Java's SceneBuilder
- The Eclipse, Netbeans, and command line were used to do the programming
- The program can run on Windows 10, Mac OS Sierra, and the latest version of Linux Ubuntu

Primary Algorithms

- All of our code is fairly straightforward logically
- Relies on action events to call methods after the main window loads
- No common algorithms used

Testing

- Mix of manual and automated testing
- Unit testing of various methods automated where possible
- Testing of beatmaker and synthesizer done manually
- Manual testing done in place of automated testing because of the steep learning curve for automating testing of GUI components

What next?

- The main things that we would like to update and improve on are
  - Performance for the music makers
  - Range of the different sounds the music makers can produce
  - Performance of the program so all users can enjoy

Demo

Questions?